CONFIGURABLE SOFTWARE SYSTEM FOR AUTOMATICALLY STORING COMPUTER FILES

Field of the Invention

This invention relates to a software application and computer interface for indexing and storing computer data files.

Background of the Invention

When storing an electronic document, a computer user typically stipulates the file name and location where the file is to be stored. The file can be retrieved at a later time either by recalling the location where the file has been stored, or if the location has been forgotten, by searching through folders, sub-folders, and files to identify the desired file. This is a user-driven task, and retrieving files is often dependent on how adept the user is at remembering file names and locations. Finding a file when the location has been forgotten is often a time consuming task. This problem is increasing as the number and size of computer files increases. Many computer systems provide the user with a search capability that enables the user to search for files under a number of criteria. Common search criteria include file name, file type, date created or modified, and file content. The number of search criteria defines the flexibility or options a user has in searching for files. Having more criteria increases a user's ability to find files, enabling a better, more precise and efficient search.

Summary of the Invention

One embodiment of the invention is directed to a process for associating scheduling data with a file consisting of the following acts: (A) stipulating a schedule of events with dates, times and activities; (B) using a computer, for example to create, modify, delete or access a current file; (C) comparing at least one characteristic of the current file against the schedule; (D) associating a searchable label to the current file when the current file matches at least one characteristic of the schedule; and (E) retrieving the current file using the label.

Brief Description of the Drawings

Fig. 1 illustrates a flow diagram of steps to carry out the filing and indexing process, according to one embodiment of the invention;

Fig. 2 illustrates a sample user interface to an application that could be used to build a schedule of activities, dates, and times;

Fig. 3 illustrates a representation of sample data associated with a schedule of classes, for example, for a student;

Fig. 4 illustrates a representation of sample data associated with a schedule of meetings, for example, for a businessperson;

Fig. 5 illustrates a representation of the process by which relevant information is accessed about a new document, according to one embodiment of the invention;

Fig. 6 illustrates an overview of the process for matching schedule attributes to a file, according to one embodiment of the invention;

Fig. 7 illustrates an overview of the process associated with a positive match between file and schedule, according to one embodiment of the invention;

Fig. 8 illustrates a database structure where attributes associated with files are stored;

Fig. 9 illustrates an overview of the process associated with a negative match between file and schedule, according to one embodiment of the invention;

Fig. 10 illustrates a user interface with which the user can search for files with specified attributes;

Fig. 11 illustrates a flow diagram of steps to carry out the filing and indexing process, according to another embodiment of the invention;

Fig. 12 illustrates a sample folder structure created through steps outlined in the process of Fig. 11;

Fig. 13 illustrates the results of identifying a positive match between file and schedule, according to the process of Fig. 11; and

Fig. 14 illustrates an overview of a process associated with a negative match between file and schedule, according to another embodiment of the invention;

Detailed Description

In accordance with one embodiment of the present invention, a configurable software system is provided for automatically assigning attributes to computer files. These attributes are stored and can be searched by a user to identify files. These attributes provide additional search criteria to the user, increasing search flexibility and enabling a better, more precise and efficient search.

A first illustrative example of the configurable software system is described making reference to Figs. 1-10. To begin, in Fig. 1, a user inputs (in act 11) a schedule into the computer system by using a user interface. A sample user interface is given in Fig. 2, although numerous other arrangements are possible. A user interface for entering a schedule may include information such as events, for example a Lecture (Fig. 2, 25); days, or date (Fig. 2, 22); times (Fig. 2, 23). In addition, related information pertaining to the event such as location, names of people associated with the event, etc., can also be captured. Schedule data can be laid out in multiple formats. Alternative data formats are shown in Fig. 3, where a mock schedule for a university student is depicted in table format; and Fig. 4, where a mock schedule for a businessperson is depicted. The nature and content of scheduling data can vary widely depending on the individual.

Once a schedule has been created, schedule data serve as a reference against which information regarding computer usage can be checked. As shown at act 12 in Fig. 1, as a user operates a computer, for example to access, modify, delete, or save files, information regarding current computer usage is accessed. This can be done in any of numerous ways. For example, current computer usage data can be derived by querying the date and time of operation. This information can reside in the computer's internal clock or operating system.

As discussed, an example of information that may be accessed through the computer's internal clock and compared against schedule data is the date and time of computer usage. An example of this process is depicted in Fig. 5, where a new file, "Acquisition Meeting Notes," (Fig. 5, 51) is being saved. The event of "Saving" the document triggers the system to access the computer's internal clock to identify current date and time. It should be noted that the event of accessing a computer's internal clock is not dependent on employing the "Save" function, or any other function in particular. Accessing of the internal clock could be repeated frequently according to a pre-set

interval, e.g., every two minutes, or triggered by a number of user events, such as opening a new file, closing a file, modifying a file, etc. Sample data accessed from the computer regarding usage, in this case, date and time regarding current usage, is depicted in Fig 5 at 52. For clarity, these data have been labeled D_{usage} and T_{usage}, respectively, where D_{usage} = September, 11, 2000; and T_{usage} = 9:30 (military time). Accessible data about current usage is not limited to the date and time.

Information about current computer usage is then compared (in act 13) to data captured in the schedule (Fig. 1). An illustrative example of this comparison is given in Fig. 6, where accessed data about usage (Fig. 6, 61), is compared against sample schedule data (Fig. 6, 62). For clarity, schedule date and time have been labeled, D_{sch} and T_{sch}, respectively. T_{sch} has two components, T_{sch1} and T_{sch2}, to capture event start time and end time (Fig. 6, 63 and Fig. 6, 64). Much more data may reside in the schedule, for example location, participants, etc., as shown in Fig. 6, 65.

The process of comparing computer usage data to schedule data may take many forms. Many different comparison means, or matching algorithms, can be employed to achieve an effective comparison. As one example of a matching algorithm, D_{usage} and T_{usage} can be compared to D_{sch} and T_{sch}. One illustrative Boolean expression for this sample algorithm is:

If $D_{usage} = D_{sch}$, and T_{usage} μ T_{sch1}, and T_{usage} [T_{sch2}, then Match = "Yes"
Else, Match = "No"

This would identify a match if the current date matched the schedule data, and the current time was within the time boundaries of an event stipulated in the schedule. As discussed, matching algorithms can take many forms. An example of an alternative algorithm is one where a modified T_{usage}, for example, T_{usage} + 10 minutes, is compared to the schedule. This algorithm could take into account discrepancies between computer usage and the start time of events, potentially improving matching accuracy. Other more complex algorithms can be employed to improve matching accuracy under different usage conditions.

An overview of this process is illustrated in Fig. 7. Date of current computer usage (Fig. 7, 71) is compared to the date of an event in the schedule (Fig. 7, 72). In

addition, the time of current computer usage (Fig. 7, 73) is compared to the times of an event in the schedule (Fig. 7, 74). Because the dates are equivalent, and the time of current usage is within the bounds of the times associated with an event in the schedule, a match is found ("Matched: YES"), (Fig. 7, 75).

Finding a successful match results in attributes from the matched event in the schedule, e.g., activity, date, time, location, participants, etc. being associated with the current file in act 15 (Fig. 1). This association can be executed automatically by the software, or by querying the user and asking for confirmation. This association can be stored within the scheduling program itself, or through other means, for example, storing attributes with files in a database. An example of this association is given in Fig. 8, where a file has been successfully associated with schedule attributes in a database structure.

If a successful match is found, additional acts, such as providing the user the opportunity to add or customize additional attributes to be associated with the file (act 16 in Fig. 1), can be incorporated into the process. Many user-definable features, such as customizing the degree to which file association is automated, can also be incorporated into the process. A further option could enable the user to cancel the association process altogether.

In the event of a negative match, (following the previous example):

$$\text{If } D_{usage} = D_{sch}\text{, and } T_{usage} \ \mu\ T_{sch1}\text{, and } T_{usage} \ [\ T_{sch2}\text{, then Match} = \text{"Yes"}$$
$$\text{Else, Match} = \text{"No"}$$

the user can be queried (in act 17 in Fig. 1) directly for attributes to save or associate with the file, and the user can define attributes (act 18). An example of this process is given in Fig. 9, where data about the current file (Fig. 9, 91) do not match data associated with events in the schedule (Fig. 9, 92). In this example, the user is prompted to associate additional attributes to the file, (Fig. 9, 93). In this example, the user chooses to define and enter a new attribute, Fig 9, 94. These data are then associated with the file. It is understood that the user may choose not to associate any attributes with a file, whether the result of a match is negative or positive, and that it is desirable that the user have some degree of flexibility in customizing the matching and file

association process.

The final act in the process relates to the retrieval of data, Fig. 1, 19. In the preceding acts, schedule attributes have been associated with files. The user is now capable of searching and identifying files with these attributes by employing a search tool (e.g., any common search tool). An illustrative example of the process and user interface for such a query is given in Fig. 10. In this example, a search window, "Find File" enables the user to search files by attributes (Fig. 10, 101). The data to be searched for is the name, "Jack Welsh" (Fig. 10, 102). The search functionality queries the database of files and associated attributes and returns the full results (Fig. 10, 103).

A second illustrative example of the configurable software system is shown in Fig. 11. To begin, in act 251, a user inputs a schedule into the system by using a user interface. A sample user interface is given in Fig. 2. As discussed previously, a user interface for entering a schedule typically includes information such as events, for example a Lecture (Fig. 2, 25); days, or date (Fig. 2, 22); times (Fig. 2, 23). In addition, related information pertaining to the event such as location, and names of people associated with the event, can also be captured (Fig. 2, 21). Schedule data can be laid out in multiple formats, alternative data formats are shown in Fig. 3, where a mock schedule for a university student is depicted in table format; and Fig. 4, where a mock schedule for a businessperson is depicted. The nature and content of scheduling data can vary widely depending on the individual.

Once a schedule has been created, a folder structure corresponding to said schedule is generated in act 252. An example of such a folder structure is given in Fig. 12. This folder structure corresponds to data shown in Fig. 3, where activities, "Physics" and "Spanish," (Fig. 3, 31), have been captured as file folders under the folder "Classes" 261 in Fig. 12.

Once a schedule and folder structure have been created, data associated with the schedule serve as a reference against which information regarding computer usage can be checked. As shown in act 253, as a user operates a computer, for example to access, modify, delete, or save files, information regarding current computer usage is accessed. This can be done in any of numerous ways as discussed above, including by querying the date and time of operation, which can reside in the computer's internal clock or operating system. As discussed previously, Fig. 5 illustrates this process.

Information about current computer usage is then compared (in act 254) to data captured in the schedule. The previously described matching process, algorithms and illustrations in Figs 6 and 7, are also applicable for this embodiment of the invention.

Finding a successful match results in the current file being saved (in act 256) in the folder associated with the matched schedule data, and attributes associated with the matched schedule data being associated with the current file. This association can be executed automatically by the software, or by querying the user and asking for confirmation. This association can be stored within the scheduling program itself, or through other means, for example, storing attributes with files in a database. An example of this saving and association is given in Fig. 13, where a file "File_Name.doc" (Fig. 27, 271) has been successfully saved in a folder, "Heinz Acquisition Meeting" (Fig. 27, 272) and associated with schedule attributes in a database structure (Fig. 27, 273).

If a successful match is found, additional acts, such as providing the user the opportunity to add or customize additional attributes to be associated with the file, could be incorporated into the process. Many user-definable features, such as customizing the degree to which file association is automated, could also be incorporated into the process. A further option could enable the user to cancel the association and filing process altogether.

In the event of a negative match, the user can be queried (act 257) directly for the target folder or save destination. An example of this process is given in Fig. 14, where data about the current file (Fig. 28, 281) do not match data associated with events in the schedule (Fig. 28, 282). In this example, the user is prompted (in act 258) to associate additional attributes to the file. In this example, the user chooses to save the file in the Miscellaneous Folder 284, a subfolder of the Activities folder. Because there is no schedule data associated with this folder, no attributes are associated with the file by the system. It is understood that the user may have options with the system to add or customize attributes according to his or her preferences.

The final act in the process relates to the retrieval of data in act 259. In the preceding acts, folders and schedule attributes have been associated with files. The user is now capable of searching and identifying files with these attributes by employing a search tool. Again, an illustrative example of the process and user interface for such a query is given in Fig. 10. In this example, a search window, "Find File" enables the user

to search files by attributes (Fig. 10, 101). The data to be searched for is the name, "Jack Welsh" (Fig. 10, 102). The search functionality queries the database of files and associated attributes and returns the full results.

Having described several embodiments of the invention in detail, various modifications and improvements will readily occur to those skilled in the art. Such modifications and improvements are intended to be within the spirit and scope of the invention. Accordingly, the foregoing description is by way of example only, and is not intended as limiting. The invention is limited only as defined by the following claims and equivalents thereto.